# 8-stage Asynchronous State Machine

SLG47921V

## Abstract

This application note explains how to design an 8-stage Asynchronous State Machine using the on-chip internal Oscillator.

This document is supported by design files listed in the Reference section.

## Contents

1. On-Chip Oscillator ..... 2
2. Ingredients ..... 4
3. Design Overview ..... 4
4. Verilog Code ..... 5
5. Floorplan & CLB Utilization ..... 9
6. Design Steps ..... 9
7. Conclusion ..... 11
8. Terms and Definitions ..... 11
9. Reference ..... 11
10. Revision History ..... 11

# 1. On-Chip Oscillator

The SLG47921V has a High-frequency on-board oscillator for use within the high-density digital FPGA Core. During either of low-power modes of the FPGA Core the high-frequency OSC is disabled. When the OSC_CTRL_EN signal is pulled HIGH, there is a delay in receiving the signal at OSC_CLK. This delay allows the signal to stabilize, hence, getting rid of any glitches in the output from the start. It has the following operating features:

- Two operating modes: high frequency (50 MHz)
- Possibility to power-down oscillator with the OSC_EN signal
- Ready signal to indicate the output clock is stable

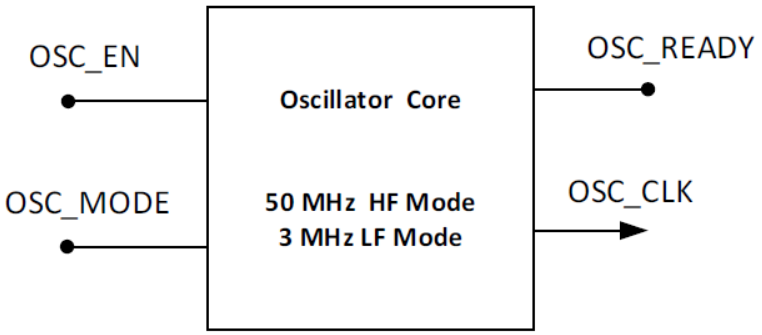


**Figure 1. High-Frequency Oscillator Block Diagram**

### Signal Descriptions

Control inputs are derived from the FPGA Core.

- OSC_EN – Active HIGH enable signal for the oscillator
- OSC_Mode – High/Low Frequency Mode selection. A HIGH-level input corresponds to 50 MHz; LOW level corresponds to 3 MHz.
- OSC_CLK – Buffered oscillator clock output. It can be a clock source for PLLs or Oscillator Post Divider. It is connected to Global Clock Distribution Network only through Post Divider
- OSC_READY – Outputs HIGH level on this signal to indicate that the oscillator frequency is stable

### Oscillator Post Divider

This circuit is intended to provide a low-frequency clock by dividing the Oscillator clock. The following functionality is implemented:

- Divide Oscillator clock by factor of N in range 1-128 (step with power of 2)
- Two divided output clocks OUT0/1 (each selected independently).
- IOB to select the division factor for each clock output individually.
- The ability to output raw Oscillator clock.
- Glitch-free and fast switching between different division factors.
- Two flags to indicate that the output clock is settled for each postdiv_out0/1

Oscillator post divider implementation is shown below.

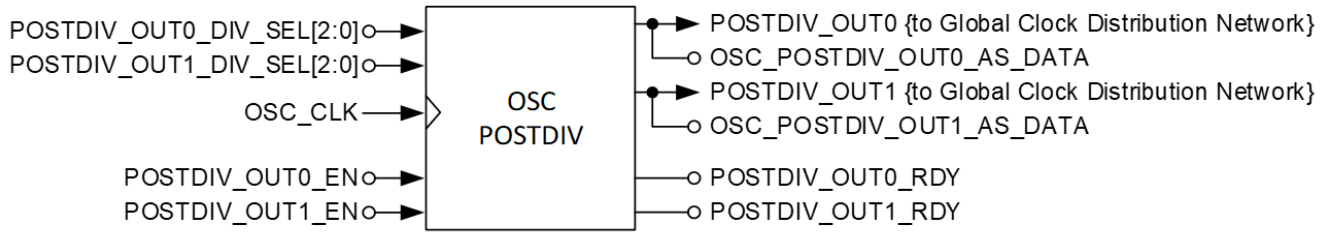


**Figure 2. Oscillator Post Divider**

Oscillator Post Divider outputs are connected to the Global Clock Distribution Network and at the same time – to dedicated IOBs of FPGA Fabric to be used as low-frequency control signals if needed.

Control inputs are derived from the FPGA Core.

- POSTDIV_OUT0/1_EN – Active HIGH enable signal for the Oscillator Post Divider Out0/1 (separate for each)
- POSTDIV_OUT0/1_DIV_SEL – signal to select division ratio for Out0/1 of Oscillator Post Divider (separate for each)
- POSTDIV_OUT0/1 – Oscillator Post Divider Out0/1 clock output. It is connected to Global Clock Distribution Network.
- OSC_POSTDIV_OUT0/1_AS_DATA – Oscillator Post Divider Out0/1 clock which is fed into FPGA Core as data signal
- POSTDIV_OUT0/1_READY – outputs HIGH level on this signal to indicate that the Oscillator Post Divider Out0/1 is stable (separate for each)
-

Oscillator Post Divider OUT0 output frequency is connected to OSC_POSTDIV_OUT0

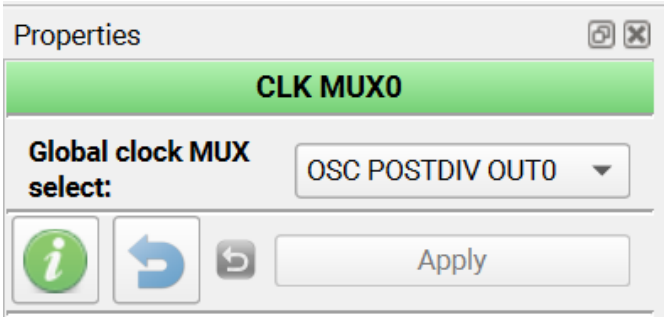


**Figure 3. Oscillator Post Divider Output**

The above figure (See **figure 3**) displays the settings for OSC Post Divider in the GoConfigure Software that needs to be set to observe the output as per the need.

Available division stages and correspondent frequencies of the Post Divider are following:

**Table 1. Oscillator Post Divider available frequencies**

| Division factor | Oscillator High Frequency Mode Output | Oscillator Low Frequency Mode Output |
|---|---|---|
| $F_{osc}$ | 50 MHz | 3 MHz |
| $F_{osc}$ /2 | 25 MHz | 1.5 MHz |
| $F_{osc}$ /4 | 12.5 MHz | 750 kHz |
| $F_{osc}$ /8 | 6.25 MHz | 375 kHz |
| $F_{osc}$/16 | 3.125 MHz | 187.5 kHz |
| $F_{osc}$ /32 | 1.56 MHz | 93.75 kHz |
| $F_{osc}$ /64 | 781.25 kHz | 46.88 kHz |
| $F_{osc}$ /128 | 390.625 kHz | 23.44 kHz |

# 2. Ingredients

- SLG47920/21V Device
- ForgeFPGA Advanced Development Board Rev2.0 with USB cable and power supply
- FPGAPAK Socket Adaptor Board #3, STQFN-48(6mm x 6mm)
- Latest Revision of ForgeFPGA Workshop software

# 3. Design Overview

In complex digital system design, state machines are widely used to model control logic. Traditional synchronous state machines rely on a global clock to trigger state transitions, which may cause timing bottlenecks and increase power consumption in high-frequency or multi-module systems. To address these limitations, the **Asynchronous State Machine (ASM)** was introduced.

Unlike synchronous designs, asynchronous state machines operate without a global clock. State transitions are triggered by events or signal changes, allowing for faster response times and lower power consumption. This design approach offers high modularity and flexibility, making it well-suited for low-power circuits, distributed systems, and high-speed data processing applications

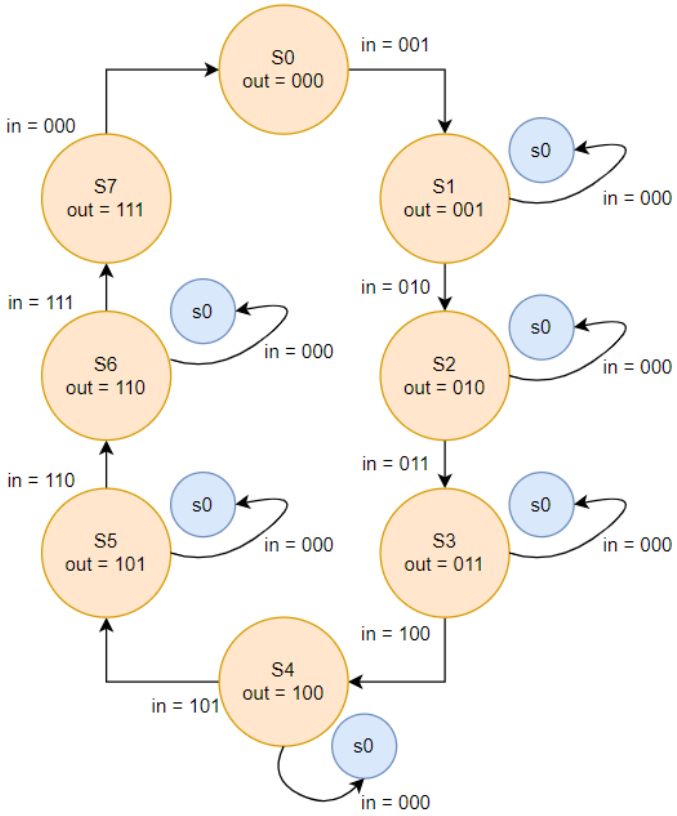


**Figure 4. State Machine Flow**

# 4. Verilog Code

Shown below is the configuration shown in the module named Counter. The Verilog code for 8 stage Asynchronous State Machine can be found in the complete design example. It is available for download (AN-FG-025 8-stage Asynchronous State Machine.ffpga)

```
(*top*) module stage8_asm (
        (*iopad_external_pin, clkbuf_inhibit*) input clk,
        (*iopad_exteranl_pin*) output osc_en,
        (*iopad_exteranl_pin*) output osc_mode,
        (*iopad_external_pin*) output o_postdiv0_ctrl_en,
        (*iopad_external_pin*) output [2:0] o_postdiv0_ctrl_sel,
        (*iopad_external_pin*) input [2:0] in,
        (*iopad_external_pin*) output reg [2:0] out,
        (*iopad_exteranl_pin*) output [2:0] out_oe
);
// register declarations
reg [7:0]       ctr = 8'd0;
reg             rstn;
reg [2:0]       current_state, next_state;
```

```
parameter       S0 = 3'd0,
                S1 = 3'd1,
                S2 = 3'd2,
                S3 = 3'd3,
                S4 = 3'd4,
                S5 = 3'd5,
                S6 = 3'd6,
                S7 = 3'd7;

// combinational output
/* Select divider value for on-chip oscillator posdivier0
3'b000 - OSC
3'b001 - OSC/2
-------------
3'b110 - OSC/164
3'b111 - OSC/128
*/
assign osc_en = 1'b1;
assign osc_mode = 1'b1;
assign o_postdiv0_ctrl_en = 1'b1;
assign o_postdiv0_ctrl_sel = 3'b000;

//generate internal reset
always@(posedge clk)
begin
        if (ctr >= 8'd50) begin
                ctr     <= ctr;
                rstn    <= 1'b1;
        end
        else begin
                ctr     <= ctr + 1;
                rstn    <= 1'b0;
        end
end
// Synchronous block for state transitions
always @(posedge clk or negedge rstn)
begin
        if (!rstn)
                current_state <= S0;
```

```
        else
                current_state     <= next_state;
end
// Combinational block for next state and output logic
assign out_oe   = 3'b111;
always @(*)
begin
        next_state = current_state;
        case (current_state)
                S0: begin
                        out       = 3'b000;
                        if (in == 3'b001)
                                next_state = S1;
                end
                S1: begin
                        out = 3'b001;
                        if (in == 3'b010)
                                next_state = S2;
                        else if (in == 3'b000)
                                next_state = S0;
                end
                S2: begin
                        out = 3'b010;
                        if (in == 3'b011)
                                next_state = S3;
                        else if (in == 3'b000)
                                next_state = S0;
                end
                S3 : begin
                        out = 3'b011;
                        if (in == 3'b100)
                                next_state = S4;
                        else if (in == 3'b000)
                                next_state = S0;
                end
                S4 : begin
                        Out = 3'b100;
                        if (in == 3'b101)
                                next_state = S5;
```

```
                else if (in == 3'b000)
                        next_state = S0;
            end

            S5 : begin
                out = 3'b101;
                if (in == 3'b110)
                        next_state = S6;
                else if (in == 3'b000)
                        next_state = S0;
            end
            S6 : begin
                out = 3'b110;
                if (in == 3'b111)
                        next_state = S7;
                else if (in == 3'b000)
                        next_state = S0;
            end
            S7 : begin
                out = 3'b111;
                if (in == 3'b000)
                        next_state = S0;
            end
            default: begin
                out= 3'b000;
                next_state = S0;
            end
        endcase
end
endmodule
```

# 5. Floorplan & CLB Utilization

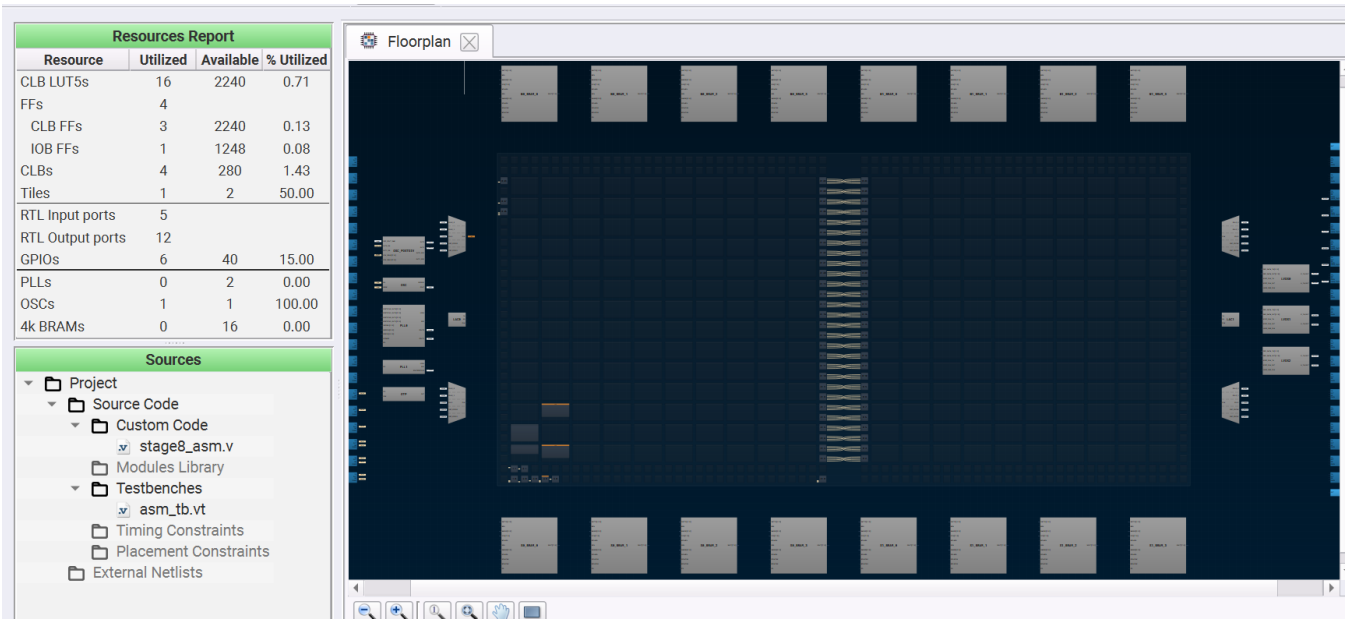


**Figure 5. 2-tile Floorplan**

The Floorplan tab in the FPGA Editor shows the placement of the CLBs, FFs and their connections to IOB blocks of SLG47921V.

Hovering over the connection tabs will highlight where the connection leads to for further understanding.
User can zoom in using the footer controls or by pressing CTRL + Scroll on the mouse.

User can observe the resources used for this design on the left top under Resources Report. User can read more details under the Resources Report tab in the toolbar.

# 6. Design Steps

1. Launch the latest version of the Go Configure Software Hub. Select the SLG47921V device and the ForgeFPGA Workshop software will load.
2. Download and open the design example AN-FG-025 8-stage Asynchronous State Machine.ffpga.
3. Open the FPGA editor and review the Verilog code. The module name is ASM which covers the working of the State Machine using internal oscillator
4. Open the IO planner tab on the FPGA editor and review the pin assignment.
5. Select the Synthesize button on the lower left side of the FPGA editor.
6. Select the Generate Bitstream button on the lower left side of the FPGA editor. Check the Logger and Issues tabs to make sure that the bit steam was generated correctly.
7. Click on the Floorplan tab and see the CLB utilization (See Figure 5). Press the Ctrl and the mouse wheel to zoom-in. Confirm that the IOs selected in the IO Planner are shown in the floorplan.
8. Launch the GTKWave Software to observe the simulation results using the testbench stimuli (See Figure 6)
9. Connect the Advanced Development Board and attach it to Adaptor Board # 3with the SLG47921V, STQFN-48 part in the socket on it. Click on the Debug button on the ForgeFPGA Workshop studio and select Emulation.

10. User can observe that each GPIO has been labeled as per their assigned functionality. User can observe the GPIO0, GPIO1 and GPIO2.
11. Launch the Logic Analyzer from the top toolbar to view waveform outputs. Inside the Logic Analyzer. Use 50Msps as sample clock rate to monitor the waveform. Press Start button from of Logic Analyzer and Pattern Generator get the GPIOs you want to observe (See Figure 7)

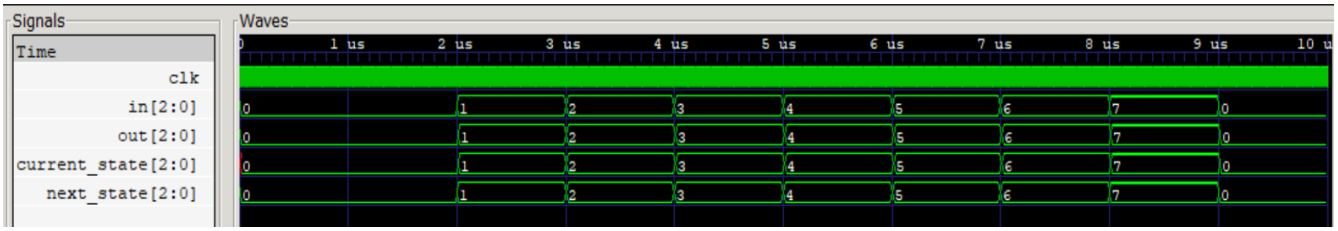


**Figure 6. Simulation Results**

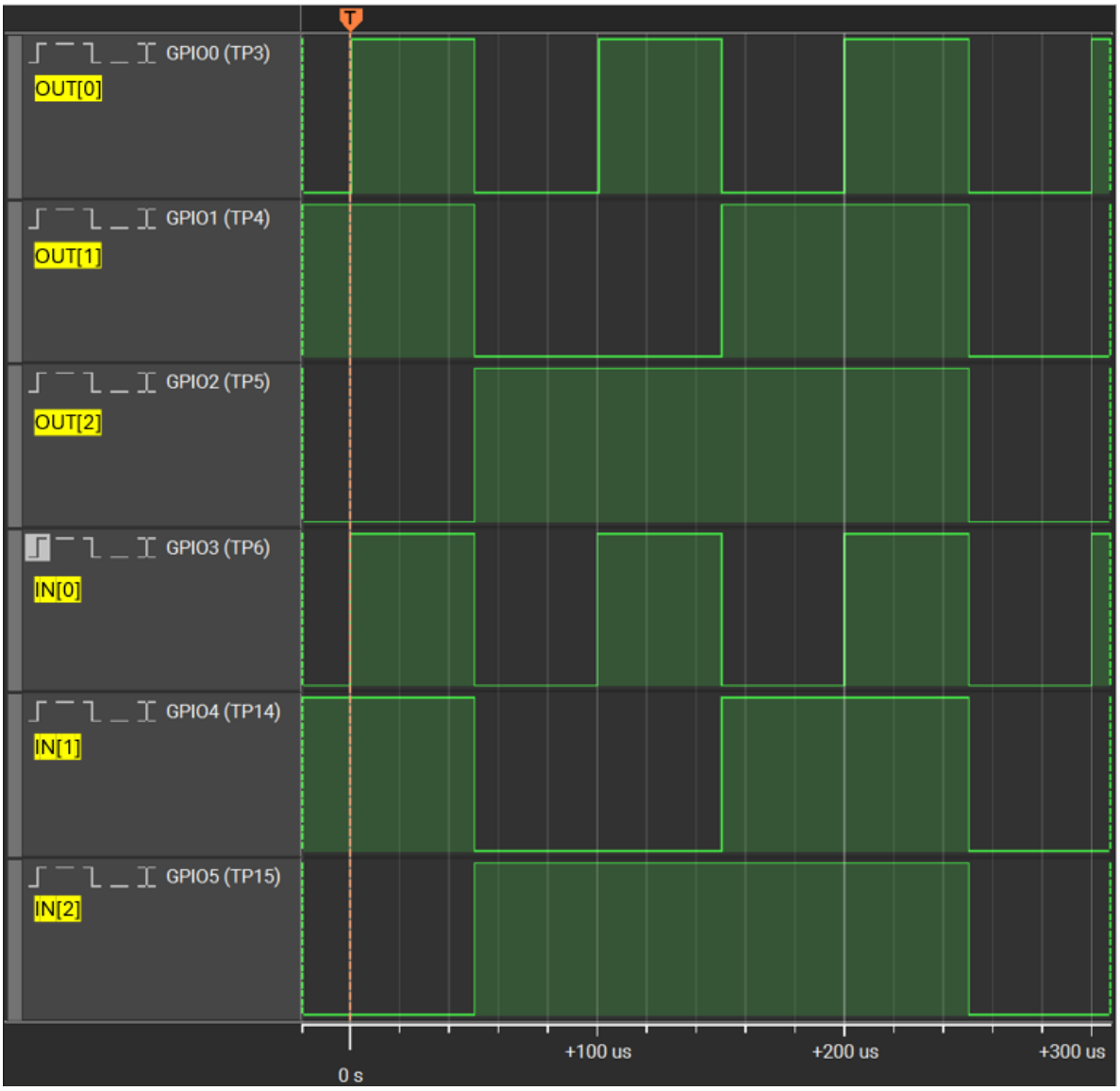


**Figure 7. Results**

# 7. Conclusion

This Application Note focuses on how to design Asynchronous State Machine on SLG47921V and how to activate it using the GoConfigure Software. The design also implements internal oscillator. This design file is available for download (AN-FG-025 Asynchronous State Machine.ffpga).

For more information, contact the ForgeFPGA Business Support Team.

# 8. Terms and Definitions

| | |
|---|---|
| CLB | Configuration Logic Block |
| HDL Editor | Workspace where Verilog code is entered |
| FPGA | Field Programmable Gate Array |
| FPGA Editor | Main FPGA design and simulation window |
| Go Configure Software Hub | Main window for device selection |
| ForgeFPGA Window | Main FPGA project window for debug and IO programming |
| OSC | Oscillator |

# 9. References

For related documents and software, please visit: ForgeFPGA. Download our free ForgeFPGA™ Designer software [1] to open the. ffpga files [2] and view the proposed circuit design.

[1] Go Configure Software Hub, Software Download and User Guide

[2] AN-FG-025 Asynchronous State Machine. ffpga, ForgeFPGA Design File

[3] ForgeFPGA SLG47920/21V, Datasheet, Renesas Electronics

[4] ForgeFPGA Workshop User Guide, Renesas Electronics

# 10. Revision History

| Revision | Date | Description |
|---|---|---|
| 1.0 | Jan 06, 2026 | Initial release. |

## Corporate Headquarters

TOYOSU FORESIA,3-2-24 Toyosu,

Koto-ku, Tokyo 135-0061, Japan

www.renesas.com

## Contact Information

For further information on a product, technology, the most up-to-data version of a document, or your nearest sales office, please visit www.renesas.com/contact-us/.